

03

조건문

목차

1. if 조건문의 기본 사용 방법
2. switch 조건문
3. 삼항 연산자
4. 짧은 초기화 조건문

if 조건문

- 조건문

```
if (<불 표현식>) {  
  
}
```

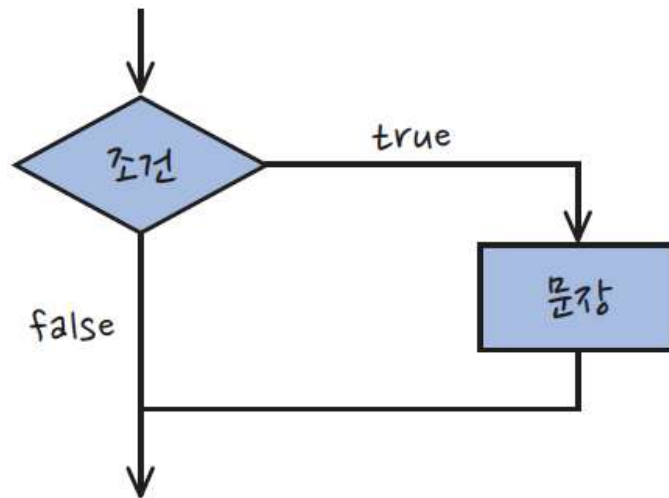


그림 3-1 if 조건문

if else 조건문

- 조건문

```
if (<불 표현식>) {  
    // 불 표현식이 참일 때 실행할 문장  
} else {  
    // 불 표현식이 거짓일 때 실행할 문장  
}
```

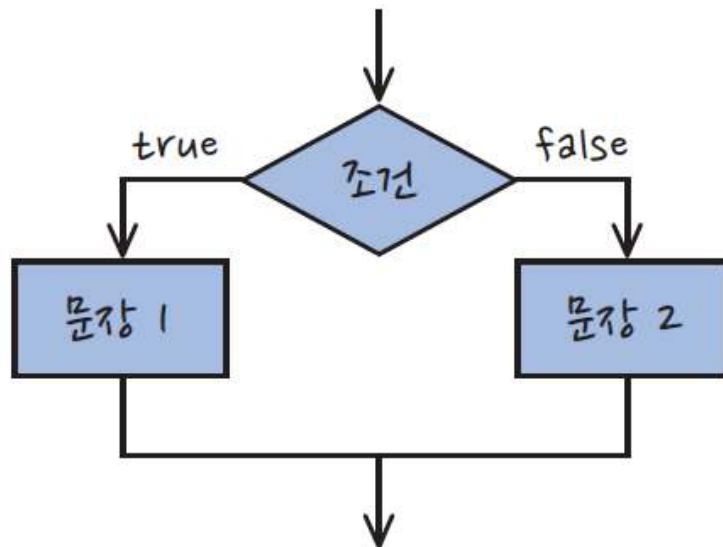


그림 3-2 if else 조건문

중첩 조건문

- 중첩 조건문

```
if (불 표현식) {  
    if (불 표현식) {  
        문장;  
    } else {  
        문장;  
    }  
} else {  
    if (불 표현식) {  
        문장;  
    } else {  
        문장;  
    }  
}
```

if else if 조건문

- 다중 조건문
 - 중복되지 않는 세 가지 이상의 조건을 구분할 때 사용

```
if (<불 표현식>) {  
  
} else if (<불 표현식>) {  
  
} else if (<불 표현식>) {  
  
} else {  
  
}
```

switch 조건문

- switch 조건문

```
switch (<비교할 값>) {  
    case <값>:  
        <문장>  
        break;  
    case <값>:  
        <문장>  
        break;  
    default:  
        <문장>  
        break;  
}
```

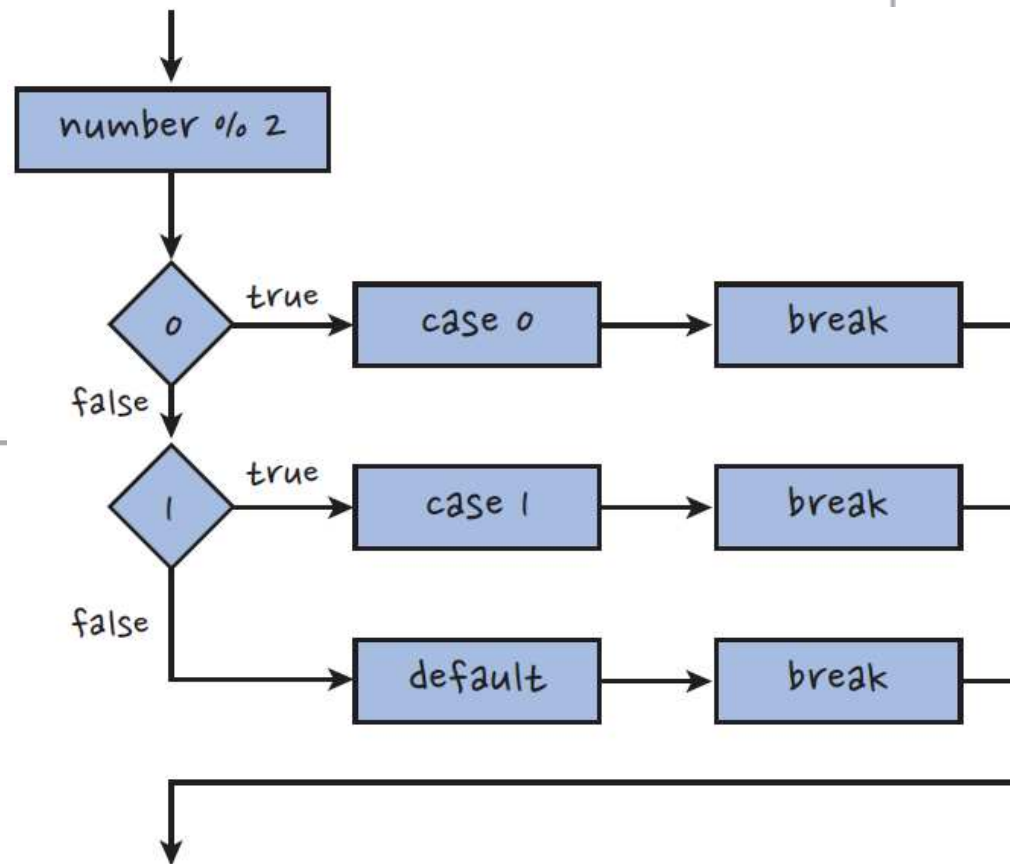


그림 3-3 switch 조건문

switch 조건문

- switch 조건문

```
const fruit = "apple";
const price = () => {
  switch (fruit) {
    case "apple": return 1000;
    case "banana": return 1500;
    case "cherry": return 2000;
    default: return 0;
  }
})();
console.log(price); // 1000
```

- return을 사용하면 break가 필요 없고, const price로 변경을 방지할 수 있습니다.

삼항 연산자

- 삼항 연산자

〈불 표현식〉 ? 〈참〉 : 〈거짓〉

```
const age = 20;
const message = `당신은 ${age >= 18 ? "성인입니다" : "미성년자입니다"}`;
console.log(message);
```

```
const user = { name: "Alice", age: 25 };

// 옵셔널 체이닝 사용 (undefined 방지)
if (user?.address?.city) {
  console.log(`도시: ${user.address.city}`);
} else {
  console.log("주소 정보 없음");
}

// 널 병합 연산자 사용 (기본값 제공)
const city = user.address?.city ?? "서울";
console.log(city); // "서울"
```

04

반복문

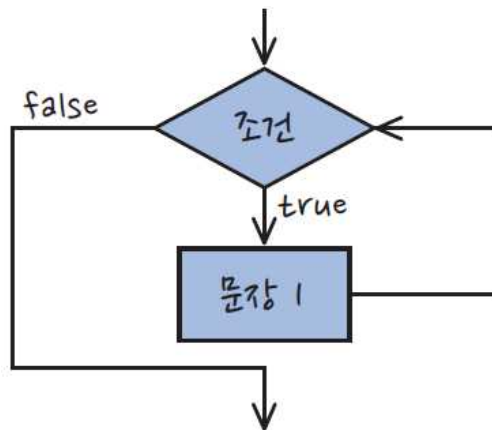
목차

1. while 반복문
2. for 반복문
3. break 키워드
4. continue 키워드

while 반복문

- while 반복문

```
while (<불 표현식>) {  
    // 불 표현식이 참인 동안 실행할 문장  
}
```

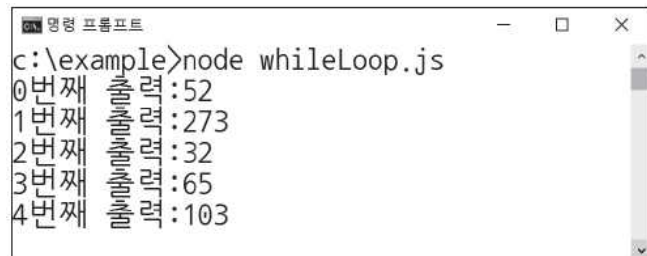


```
while (true) {  
    console.log("무한 반복");  
}
```

while 반복문

- while 반복문

```
// 변수를 선언합니다.  
let i = 0;  
let array = [52, 273, 32, 65, 103];  
  
// 반복을 수행합니다.  
while (i < array.length) {  
    // 출력합니다.  
    console.log(i + "번째 출력:" + array[i]);  
  
    // 탈출하려고 변수를 더합니다.  
    i++;  
}
```



```
명령 프롬프트  
c:\example>node whileLoop.js  
0번째 출력:52  
1번째 출력:273  
2번째 출력:32  
3번째 출력:65  
4번째 출력:103
```

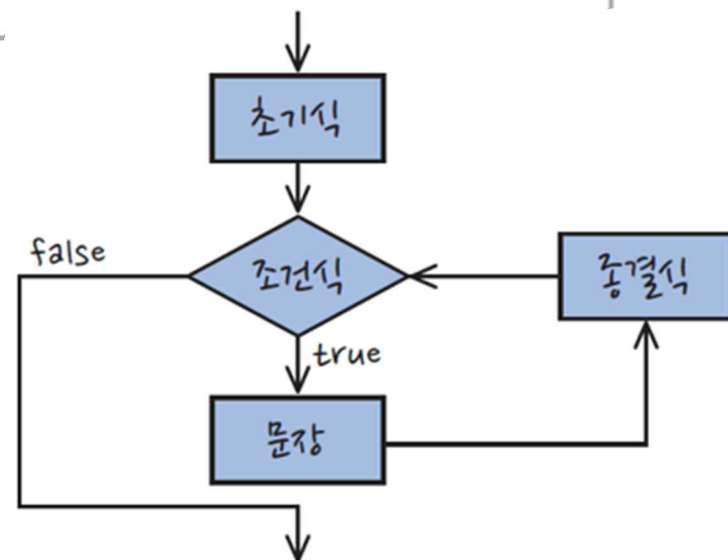
for 반복문

● for 반복문

```
for (let i = 0; i < <반복 횟수>; i++) {  
  
}
```

```
for (let i = 0; i < 배열.길이; i++) {  
    let 인덱스 = i;  
    let 요소 = 배열[i];  
}
```

```
// 배열을 생성합니다.  
let array = [1, 2, 3, 4, 5, 6];  
  
// 요소의 길이를 출력합니다.  
for (let i = array.length - 1; i >= 0; i--) {  
    console.log(array[i]);  
}
```



for in 반복문과 for of 반복문

● for in , for of 반복문

- for...in은 객체의 열거 가능한 속성(key)을 반복(iterate)하는 데 사용됩니다.
- 배열, 객체, 문자열 등 모든 열거 가능한 속성에 대해 반복하며, **key(속성 이름)에 접근**합니다
- for...of는 이터러블(iterable)한 객체를 반복(iterate)하는 데 사용됩니다.
- 배열, 문자열, Map, Set 등 **iterable 객체의 값에 직접 접근**합니다.

```
for (let 인덱스 in 배열) {  
  
}
```

```
for (let 요소 of 배열) {  
  
}
```

```
const text = "Hello";  
for (const char of text) {  
  console.log(char);  
}
```

for in 반복문과 for of 반복문

- for in , for of 반복문

```
// 변수를 선언합니다.
let array = ["사과", "배", "포도", "딸기", "바나나"];

// 반복을 수행합니다.
for (let i in array) {
    // 출력합니다.
    console.log(`${i}번째 요소: ${array[i]}`);
}

console.log("----- 구분선 -----");

// 반복을 수행합니다.
for (let item of array) {
    // 출력합니다.
    console.log(item);
}
```


for in 반복문과 for of 반복문

- `forEach()`에서는 `return`을 써도 반복을 종료할 수 없습니다.
- `forEach()`는 `break`나 `continue`를 사용할 수 없습니다.
- `some()`은 `true`를 반환하면 반복을 중단합니다.

```
const numbers = [1, 2, 3, 4, 5];

numbers.some(num => {
  console.log(num);
  return num === 3; // 3에서 반복 중단
});
```

```
const numbers = [1, 2, 3, 4, 5];

for (const num of numbers) {
  if (num === 3) break;
  console.log(num);
}
```

for in 반복문과 for of 반복문

- 조건문 및 반복문에서 비구조화 할당 (디스트럭처링)을 사용

```
const users = [  
  { name: "Alice", age: 25 },  
  { name: "Bob", age: 30 }  
];  
  
for (const { name, age } of users) {  
  console.log(`${name}의 나이는 ${age}세입니다.`);  
}
```

- for...in은 객체 속성을 반복할 때, 상속된 속성까지 포함하는 문제는 Object.hasOwn()을 사용하여 해결할 수 있습니다.

```
const user = { name: "Alice", age: 25 };  
  
for (const key in user) {  
  if (Object.hasOwn(user, key)) {  
    console.log(`${key}: ${user[key]}`);  
  }  
}
```

for in 반복문과 for of 반복문

- ES8(ES2017)에서 도입된 async/await를 활용하면 비동기 루프를 더 쉽게 구현
- for-await-of를 사용하여 순차적으로 처리 [Promise.all() 대체]

```
const fetchData = async () => {  
  return new Promise(resolve => setTimeout(() => resolve("데이터 로드 완료"),  
    1000));  
};  
  
const asyncProcess = async () => {  
  const promises = [fetchData(), fetchData(), fetchData()];  
  
  for await (const result of promises) {  
    console.log(result);  
  }  
};  
  
asyncProcess();
```

중첩 반복문

- 중첩 반복문

```
let output = "";

for (let i = 0; i < 10; i++) {
  for (let j = 0; j < i + 1; j++) {
    output += '*';
  }
  output += '\n';
}

console.log(output);
```

break 키워드

- 무한 루프와 break

- 무한 반복문은 내부에서 반드시 종료조건과 break 키워드를 포함해야 합니다

```
while (true) {  
  
}
```

```
let i = 0;  
let array = [1, 31, 273, 57, 8, 11, 32];  
let output;  
  
while (true) {  
  if (array[i] % 2 == 0) {  
    output = array[i];  
    break;  
  }  
  
  i = i + 1;  
}  
  
console.log(`처음 발견한 짝수는 ${output}입니다.`)
```

continue 키워드

- continue

- 반복문 내부에서 현재 반복을 멈추고 다음 반복을 진행함

```
for (let i = 1; i < 10; i++) {  
  if (i % 2 == 0) {  
    continue;  
  }  
  
  console.log(i)  
}
```

짝수라면 다음 반복으로 바로 넘어갑니다.
따라서 이 다음 코드는 실행되지 않습니다.

연습 문제

- 실습문제

Q1. 사용자에게 숫자를 입력받아 짝수인지 홀수인지 출력하는 로직을 작성하세요.

Q2. 2부터 9까지의 구구단을 단별 가로로 출력하는 로직을 작성하세요.